NEXT.JS #2

레이아웃 템플릿 / 라이프사이클 / 컴포넌트

// _document.js //

- /pages/_document.js 에 생성
- 기본 Document 클래스를 확장하는 커스텀 클래스 형태로 제작
- 웹 사이트 전체를 감싸는 템플릿 역할 수행
- <head> 태그 내에 필요한 내용들을 선언할 수 있음
- Head, Main, NextScript 컴포넌트를 반드시 포함해야 함

// <Head> //

- _document 를 생성할 때 가장 유용하게 사용되는 기본 컴포넌트
- _document 안에서 선언시에는 'next/document' 패키지 사용
  import { Head } from 'next/document';
- 다른 페이지에서 사용시에는 'next/head 패키지 사용
  import Head from 'next/head'
- _document 에서 선언되면, 각 페이지에서 부분 재정의 가능

// 기본 형태 //

```
import Document, { Head, Main, NextScript } from 'next/document';

export default class CustomDocument extends Document {
  render() {
    return (
      <html>
        <Head>
          ...
        </Head>
        <body>
          <Main/>
          <NextScript/>
        </body>
      </html>
    );
  }
}
```

// Head 재정의 //

```
// Home.js

import Head from 'next/head';

class Home extends Component {
  render() {
    return (
      <div>
        <Head><title> 페이지 제목</Title></Head>
        ...
      </div>
    );
  }
}
```

// Head 에서 재정의 하는 것들//

- SEO를 위한 title, meta 태그
- open graph 태그
- 특정 페이지에서만 필요한 스크립트 혹은 스타일시트

// 라이프 사이클 //

- 생명 주기
  - 처음 생성되어 메모리에 올라가고, 화면에 표시되고, 표시가 제거된 후, 메모리에서 사라지는 일련의 과정

- 라이프 사이클 중간 중간의 특정 이벤트를 활용

“Render Phase”

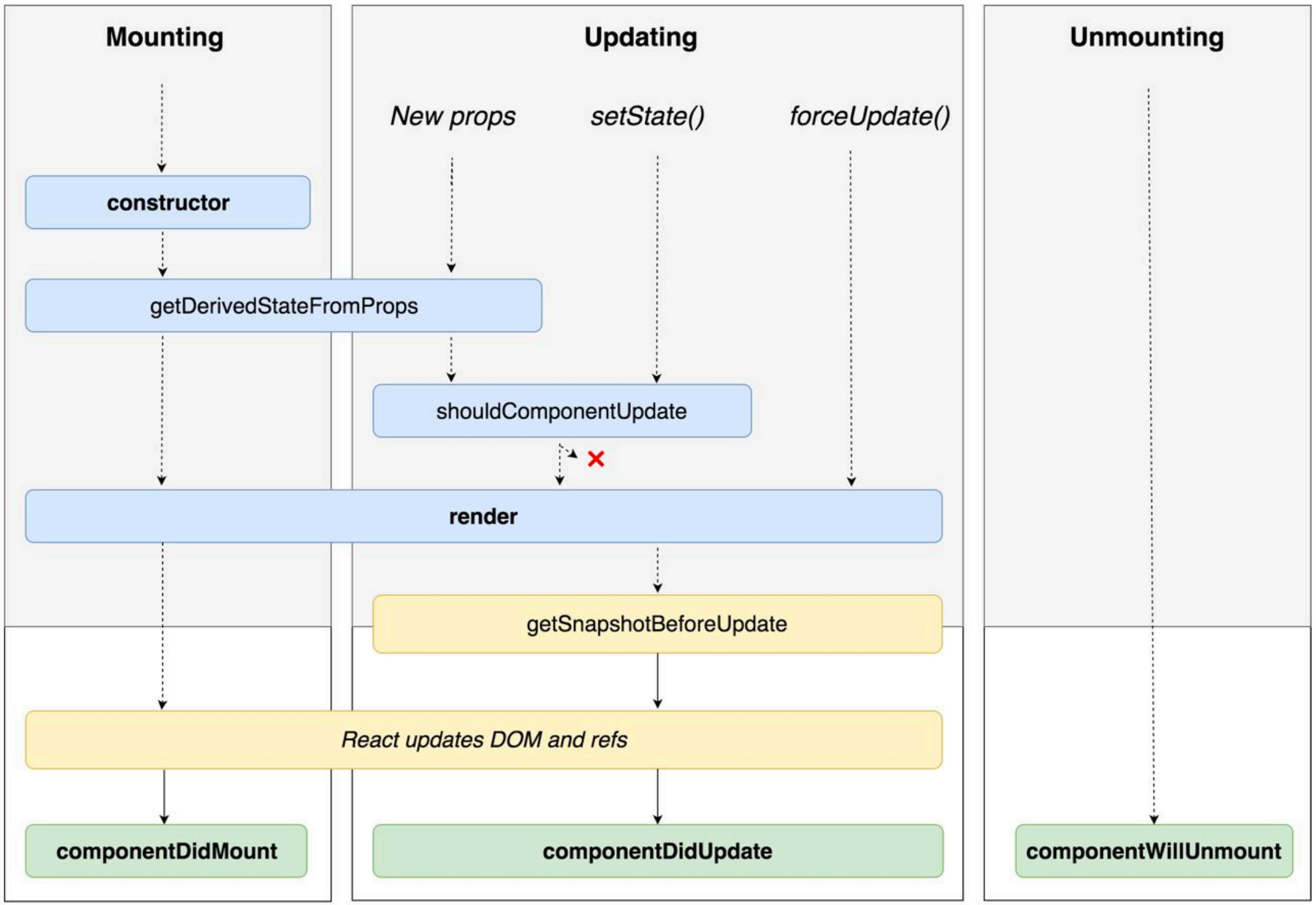
Pure and has no side effects. May be paused, aborted or restarted by React.

“Pre-Commit Phase”

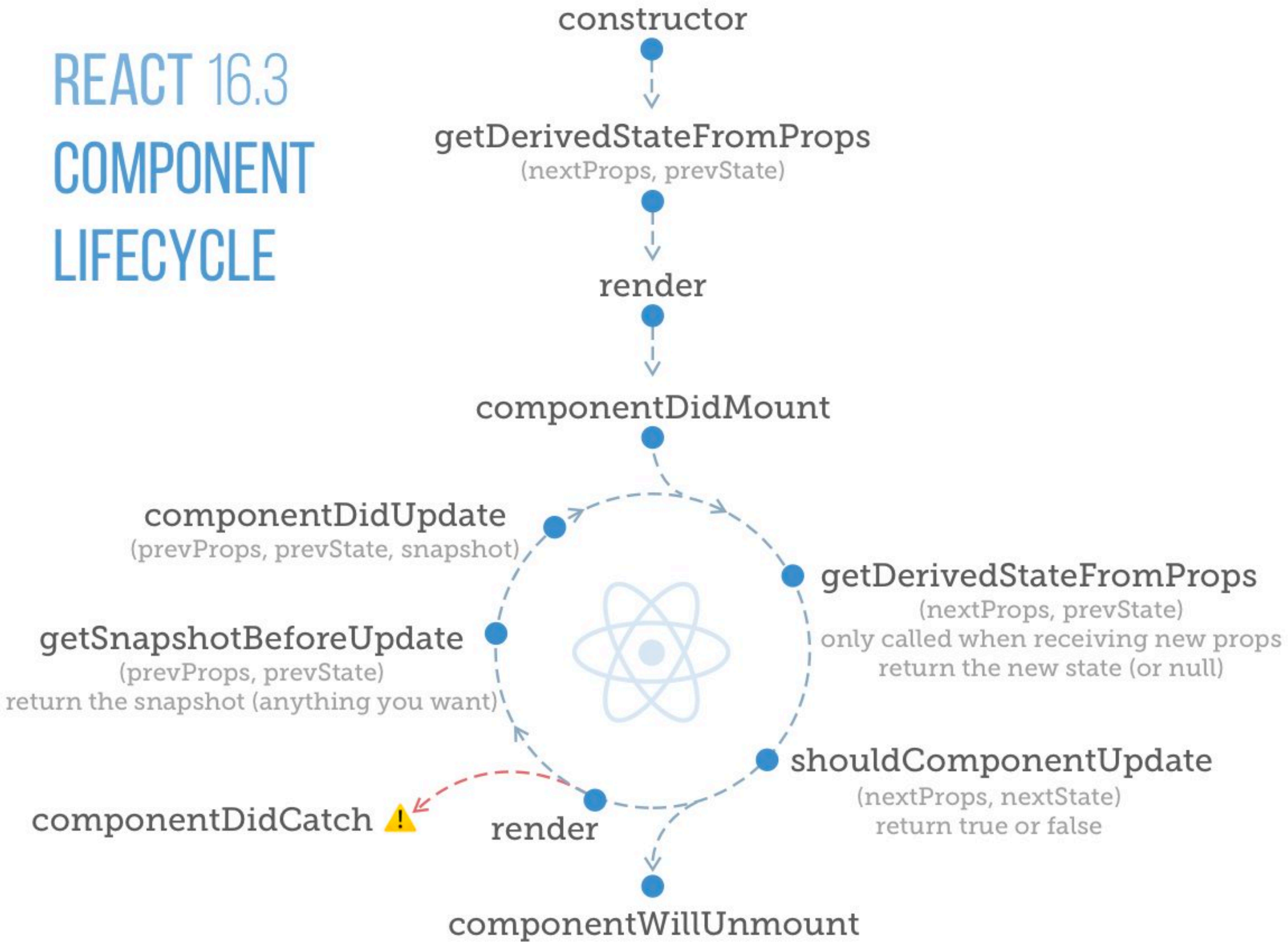
Can read the DOM.

“Commit Phase”

Can work with DOM, run side effects, schedule updates.



REACT 16.3 COMPONENT LIFECYCLE



// 주요 이벤트 //

- constructor
  객체가 생성되는 순간

- componentDidMount
  요소가 화면에 표시된 직후

- shouldComponentUpdate
  업데이트 할 요소가 있을 때 (렌더 하기 직전)

- componentWillUnmount
  화면에서 제거되기 직전

// getInitialProps //

- next.js 에만 있는 라이프 사이클
- pages 디렉토리 하위의 클래스에서만 사용 가능
- 백엔드에서 동작하는 이벤트
- 이 시점에서 API 등을 이용해 데이터를 가져오면 서버사이드 렌더링 결과에 반영 가능
- static 으로 선언해야 하며 async 함수로 선언해야 함
- 이 함수가 return 하는 오브젝트가 this.props 가 됨

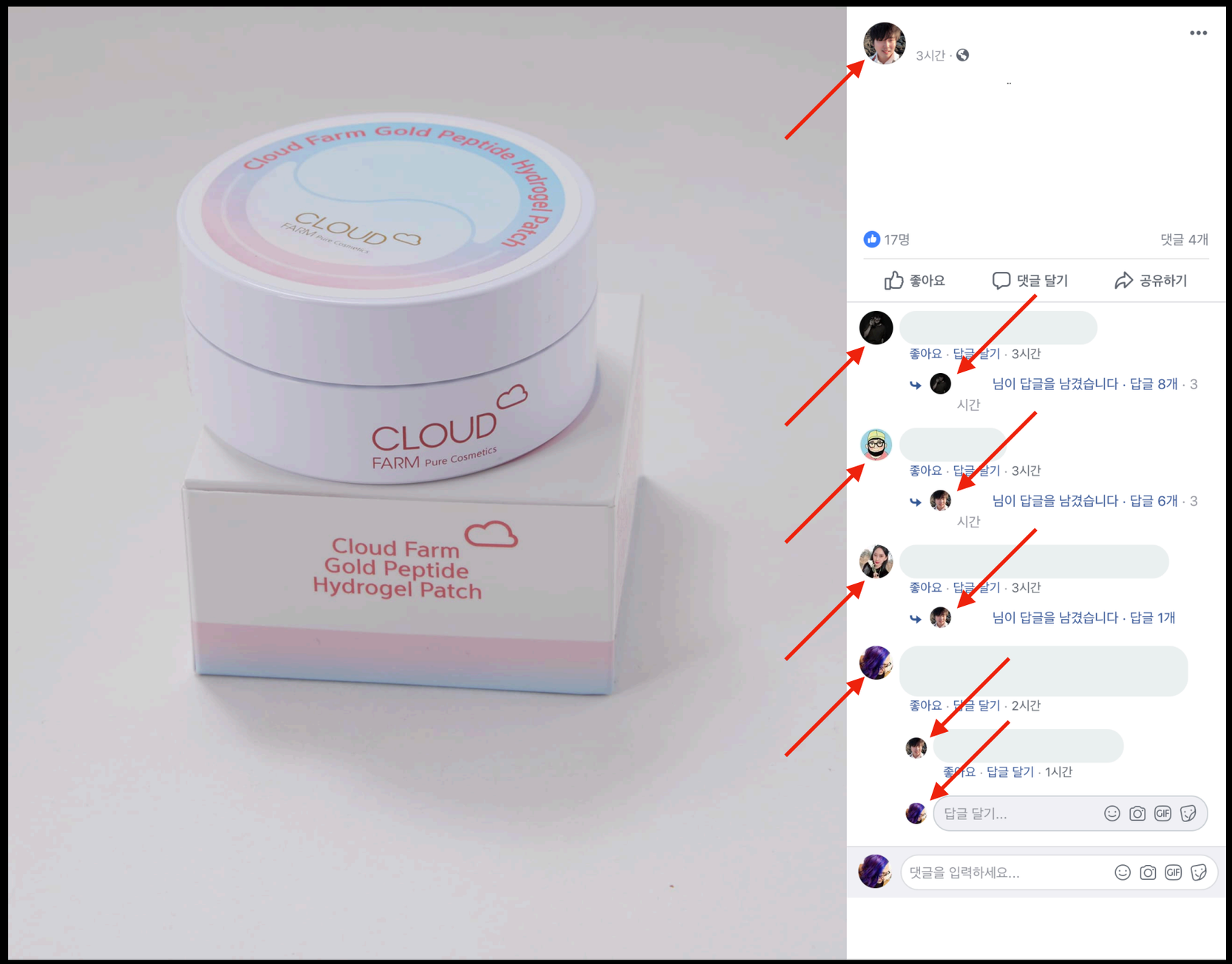
// 사용 예 //

```
class Custom extends Component {
  static async getInitialProps({ req }) {
    let userAgent = '';
    if( req ) { // http request 가 존재하는지 여부
      userAgent = req.headers['user-agent'];
    }
    else {
      userAgent = navigator.userAgent;
    }
    return { userAgent }
  }
  render() {
    return <div>Hello, { this.props.userAgent }</div>
  }
}
```

// 커스텀 컴포넌트 //

- 반복적으로 사용되는 요소를 컴포넌트로 만들어서 분리
- 코드 간결화, 높은 재사용성, 수정 용이성 증가
- props 활용: 부모 요소가 자식 요소에게 넘겨주는 데이터

// 커스텀 컴포넌트 //


Cloud Farm Gold Peptide Hydrogel Patch
CLOUD
FARM Pure Cosmetics
Cloud Farm
Gold Peptide
Hydrogel Patch
3시간 ·
17명
댓글 4개
좋아요
댓글 달기
공유하기
좋아요 · 답글 달기 · 3시간
님이 답글을 남겼습니다 · 답글 8개 · 3시간
좋아요 · 답글 달기 · 3시간
님이 답글을 남겼습니다 · 답글 6개 · 3시간
좋아요 · 답글 달기 · 3시간
님이 답글을 남겼습니다 · 답글 1개
좋아요 · 답글 달기 · 2시간
좋아요 · 답글 달기 · 1시간
답글 달기...
댓글을 입력하세요...

// props //

- HTML의 attribute 와 같은 형태
- 문자열로 전달할 수도 있고, 데이터를 바로 전달할 수도 있다
- 문자열을 전달시엔 쌍따옴표(") 를 이용
- 데이터를 전달할 땐 중괄호({}) 를 이용

// 사용 형태 //

```
// props 가 없이 쓰임
<ProfileImage/>

// size 라는 props 로 "100" 전달
<ProfileImage size="100"/>

// 숫자 형식을 그대로 넘기고 싶다면 { } 이용
<ProfileImage size={ 100 }/>

<ProfileImage url="..."/>
<ProfileImage url={ "..." }/>
```

// 컴포넌트 만들기 //

```
class ProfileImage extends Component {
  state = {}
  constructor( props ) {
    super(props);
    this.state.width = this.state.height = props.size || 200;
    this.state.url = props.url || 'https://placeimg.com/200/200/animals';
  }
  render() {
    const style = {
      width: this.state.width,
      height: this.state.height,
      backgroundImage: `url( ${this.state.url} )`,
      ...
    }
    return (
      <span style={ style }/>
    );
  }
}
```

// 컴포넌트 사용하기 //

```
import ProfileImage from '../components/ProfileImage';

class Page extends Component {
 render() {
    return (
      <div>
        <ProfileImage/>
        <ProfileImage size="150"/>
        <ProfileImage size="100"/>
        <ProfileImage url="https://placeimg.com/300/300/animals"/>
        <ProfileImage url="https://placeimg.com/400/400/animals"/>
        <ProfileImage url="https://placeimg.com/500/500/animals"/>
      </div>
    );
  }
}
```

// 질문 / 답변 / github //

질문 & 답변

코드는: github.com/GrotesQ/CodeLab-Next-JS

감사합니다.